In the United States Patent and Trademark Office

5a
Ach
4-3-02

Serial number 09/340,172
Application filed June 25, 1999
Applicant Derek Wong
Application Title Methods for Increasing Instruction-Level Parallelism in Microprocessors and Digital Systems
Examiner/GAU Eric Coleman / 2183

Mailed March 20, 2002
at San Jose, California

From:
Derek Wong
1341 Echo Valley Dr.
San Jose, CA 95120

To:
Assistant Commissioner for Patents
United States Patent and Trademark Office
Washington, DC 20231

RE: Amendment A to patent application 09/340,172

Thank you for reviewing patent application 09/340,172.

In response to the office action letter regarding patent application 09/340,172 mailed on September 20, 2001, the applicant encloses a new claim set, revised figures to replace the original figures, and this letter.

The figures have been corrected as suggested by the patent office draftsperson.

The applicant requests that claims 1-43 in the original application be cancelled and replaced with a new claim set consisting of claims 44-86 enclosed with this amendment. The applicant respectfully submits this new set of claims for examination. The new claims 44-86 correspond to the original claims 1-43 with extensive changes.

In general, many of the claims have been changed with added explanatory text and limitations to make it more defined what is being claimed as suggested by the examiner.

While examining this new claim set, the applicant also wishes the patent office to consider that the field of digital processor design has many thousands of engineers and researchers spending a total of multiple billions of dollars per year on design. The large amount of work done in this field should weigh as a factor in determinations of obviousness, i.e. if prior art does not contain some element of the claimed invention or does not suggest a combination of references, this should help weigh in favor of it being reasonable to consider the claimed invention non-obvious and patentable considering that the prior art in this field represents the thinking of a very large number of persons skilled in this art. (In other words, if something is considered obvious, why hasn't someone done it in this way already in the prior art?)

The applicant has made effort to respond to each of the comments received in the office action below.

In regard to the rejection on the basis of Hammond (US Patent 5,638,525) in the office action, the applicant has examined Hammond carefully and changed claim 1 of this application in a very substantive way to create new claim 44.

Most of the figures in Hammond do not present a system which stores translated instructions into a cache except Figure 5. The figures other than Figure 5 all translate the instructions between the cache and the execution unit, so that the translated instructions are never stored in the cache.

In Figure 5, Hammond presents a system for translating instructions from a first instruction set architecture to a second instruction set architecture using translator 541 and subsequently storing the translated instructions in instruction cache 542. An important limitation in Figure 5 of Hammond is that the execution unit in Figure 5 can only execute instructions in the second (translated) instruction set architecture. Hence there is no additional regular cache to hold instructions in the first instruction set architecture.

In contrast, the design shown in Figure 1 of the present application uses both a regular cache for instructions in the original instruction set architecture and an instruction stream cache for instructions in the transformed instruction set architecture. This allows the possibility for an execute unit that can execute instructions from both the original and transformed instruction set architectures. This dual cache architecture that can store both types of instructions is more flexible and does not require the execute unit to wait until instructions have been transformed before executing them.

Claim 44 of the present application now includes the limitation that the processor device should contain both a regular cache for instructions in the original instruction set architecture as well as an instruction stream cache for instructions in the transformed instruction set architecture. The regular cache is not an element in Hammond because Hammond's design does not have an execute unit that can execute instructions from both instruction set architectures.

Claim 44 also contains the wherein limitation that the instruction stream cache can be addressed by fetch requests by the execute unit and can potentially respond to fetch requests without

requiring cache hit information from the regular cache. This clearly distinguishes the claimed invention from Ireton (US Patent 5,826,089, column 12, lines 30-45) as explained below.

In Ireton, the fetch mechanism of the execute unit addresses the instruction cache and fetches instructions from the instruction cache. (The instruction cache in Ireton appears about equivalent to the present application's regular cache.) These instructions are then conveyed to the translation unit from the instruction cache. As these fetched instructions are conveyed to the translation unit, Ireton suggests that the translation unit can search an internal translation cache to see if the fetched instructions have already been translated, in which case the translated instructions can be provided from the translation cache.

The translation cache in Ireton does not respond directly to fetches by the execute unit. Instead, fetches from the execute unit are made from the instruction cache and then the translation cache is searched when incoming instructions are conveyed from the instruction cache.

This is clearly different than the applicant's claimed invention in claim 44, where the limitation is made that the instruction stream cache is addressed by and can potentially respond to some of the fetch requests by said execute unit for transformed code blocks without requiring cache hit information from the regular cache after said code blocks are already stored in the instruction stream cache. This still allows the possibility for fetches to simultaneously address both the regular cache and the instruction stream cache if desired to maximize speed, but in the case where the instruction stream cache provides the cache hit, then the cache hit information from the regular cache is not required (unlike in Ireton).

The applicant respectfully submits that claim 44 should be allowable.

Hammond does not present much detail on the mechanism of the translator unit. The present application does present many details of the mechanism in the instruction stream transformation unit. Many of these details are included in limitations in the subsequent claims dependent on claim 44.

Comments on claim 2: New version submitted as claim 45. This limitation is an extra restriction along the lines of the above explanation. Claim 44 should independently be allowable for the reasons above because the presence of the regular instruction cache is a major difference from Hammond.

Comments on claims 3, 4, 12, 20, and 21: New version submitted as claims 46, 47, 55, 63, 64. Some of these claims have been changed with additional explanatory text and are also dependent on new claim 44.

The examiner states that the x86 instruction set has been implemented in prior art using mechanisms that allow in-order and out-of-order execution. However, the prior art uses predominantly mechanisms for dynamically scheduling instructions just prior to being submitted to execute unit(s). The prior art does not use a mechanism as stated in claim 44 where code blocks are transformed by an instruction stream transformation unit and subsequently stored in a specialized instruction stream cache, possibly in an out-of-order fashion.

The applicant respectfully submits that claims 46, 47, 55, 63, and 64 should be allowable.

Comments on claim 5: New version submitted as claim 48. This claim has been changed to describe in more detail the function of the working memory. The applicant notes that the memory in Figure 5 of Hammond is a main memory containing instructions and data and is not used to store intermediate calculations of the transformation process. The applicant respectfully submits that claim 48 should be allowable.

Comments on claims 16-17: New version submitted as claims 49-50. The instruction window and overlapping instruction window concept is quite different from the concept of register windows cited in the office action. Page 29 of the specification describes the instruction window and overlapping instruction window concepts. These instruction windows correspond to a group of instructions that the instruction stream transformation unit will process together, rather than a window of data registers. Claims 49 and 50 have been changed with more explanatory text to describe this. The applicant respectfully submits that claims 49 and 50 should be allowable in light of the above clarification.

Comments on claims in view of Hammond and Farber (US Patent 6,105,124):

New versions of claims 6-11 have been submitted as claims 49-54.

In part because claim 44 has been changed very substantially, the combination of Hammond and Farber does not yield the claimed inventions in claims 49 and 50.

Claims 51 through 54 (new versions of old claims 8-11) in the present application cover a method used by the instruction stream cache to store and access hyper-blocks as described on pages 18 and 19 of this application. Hyper-block ID's are used to access the instruction stream

cache. The cache lines corresponding to a hyper-block are indexed by hyper-block ID and cache line number. The hyper-block ID plus line number is an inventive way to enable fetching the lines of a hyper-block in a sequence, allowing the cache to enable access to a conceptual structure called a hyper-block chain which is the list of cache lines corresponding to a hyper-block. As noted on page 19, one of the benefits of this mechanism is flexible enough to enable the cache to store multiple hyper-blocks that overlap in memory in the original instruction set architecture code.

Differences compared to Farber include the following:

- Farber does not describe mechanisms to address cache lines.
- Farber does not use identifiers for hyper-blocks. Farber does mention the use of identifiers in column 4; however, these identifiers are for basic blocks and not for hyper-blocks. A basic block identifier in Farber is used to match against identifiers stored into the Translated Address Table to find the address of a translated basic block. This does not serve the same function that hyper-block ID's do in the applicant's invention.

The applicant respectfully submits that claims 49-54 should be allowable based on the new claim 44 and in light of the above distinctions.

Comment on claim 13: New version submitted as claim 56. Predication if-conversion is presented in pages 25-27 of the specification. In this concept, an if-then statement is converted to a predicate calculation and one or more predicated instructions. Predicated instructions conditionally commit results depending on the values of the predicate(s) on which they depend. This concept is different from converting an if-then instruction into opcode/operand as the office

action describes. Claim 56 contains new additional explanatory text. The applicant respectfully submits that the limitations of claim 56 (being also dependent on claim 44's limitations) constitute an allowable claim.

Comments on claims 22-24:

- New version submitted as claims 65-67. Claims 65-67 are dependent upon claim 44 which has new content with intent to be allowable.
- Claims 66 and 67 have additional explanatory text compared to claims 23 and 24 about list scheduling and dynamic memory disambiguation.
- The algorithms in Farber do not correspond to the list scheduling algorithm described in claim 66; list scheduling is described on pages 35-39 of the specification.
- The content of claim 67, dynamic memory disambiguation, is also not discussed in Farber. Dynamic memory disambiguation is discussed on pages 39-41 of the specification.

The applicant respectfully submits that claims 66 and 67 should therefore be allowable in light of the above distinctions.

Comments on claim 27:

New version submitted as claim 70.

The applicant respectfully submits that the limitations of claim 70, being also dependent on new claim 44's limitations, constitute an allowable claim.

Comment on claim 32:

New version submitted as claim 75.

Farber does not describe the applicant's idea of semi-dynamic instruction code re-writing and re-scheduling to further optimize code based on run-time information. In this concept, already scheduled code is re-written and re-scheduled using run-time information. This idea is described on pages 44-47 of the specification including many details of some particular run-time history tables. Farber only describes collecting run-time information and then using it to schedule the code once. Claim 75 has new additional explanatory text. The applicant respectfully submits that claim 75 should be allowable in light of the above explanation.

Comments on claims in view of Ireton (US Patent 5,826,089):

As newly changed, the limitations of claim 44 are now clearly distinct from Ireton's design as explained earlier.

The applicant respectfully submits that new claim 44 and dependent claims should now be allowable. Further comments on some specific claims in view of Ireton are below.

Comment on claim 14: New version submitted as claim 57. Claim 57 describes a method of using the concept in claim 44 with the additional concept of converting a load instruction into a speculative load and a load activation pair. This allows the load to be scheduled earlier. This concept is described on pages 27-28 of the specification. In contrast, Ireton describes a mechanism where an instruction can be speculatively fetched or speculatively executed prior to a

decision being made as to whether to nullify the effects of the instruction. This is a different concept and not as powerful for scheduling loads as compared to converting load instructions into two separately schedulable instructions to allow early scheduling of the load. Claim 57 contains new additional explanatory text. The applicant respectfully submits that new claim 57 (being also dependent upon new claim 44) should be allowable.

Comment on claims 15, 18, and 19: New versions submitted as claims 58, 61, and 62. The concepts in claims 61 and 62 are described starting from the bottom of page 29 through page 33 of the specification. The cited references do not describe the creation of a dependency matrix or operand mapping table as in claim 61 or 62, and the applicant wishes to make the case that it is not an obvious extension of the cited prior art. The cited prior art has a method for handling dependencies but it is not an obvious extension to create a dependency matrix and operand mapping table as described by the applicant. The advantage of the dependency matrix and operand mapping table is very substantial in that this is a very systematic way of representing dependencies that enables scheduling to occur on large sequences of instructions at high speed. This representation also enables sophisticated algorithms that optimize over large windows of instructions such as the list scheduling algorithm described as an example scheduling algorithm in the specification.

Prior-art mechanisms use different representations and appear to optimize scheduling over smaller groups of instructions. Prior-art dynamically-scheduled execute units may select instructions for execute based on heuristics such as prioritizing among ready instructions (those that have all dependencies resolved) based on a metric such as how long the instruction has been

waiting to execute. This is essentially a version of a first-come, first-served algorithm based on a metric. This is much less sophisticated than the list scheduling algorithm.

In summary, the described dependency matrix and operand mapping table mechanisms constitute a significant advance that enable scheduling algorithms to optimize over large groups of instructions more efficiently and quickly. If the claimed inventions were obvious extensions of the prior art, then the substantial advantages of the dependency matrix and operand mapping table should have strongly encouraged others to develop this in prior art. Claims 58, 61, and 62 contain new additional explanatory text and are dependent upon new claim 44. The applicant respectfully submits that these claims 58, 61, and 62 should be allowable in view of the above reasons.

Comment on claim 40: New version submitted as claim 73. Some of the significant benefits of a dependency matrix have been described above, and such a structure is not present in Ireton. For a software scheduler, this data representation should enable the scheduling algorithm to execute much faster than using a directed-acyclic-graph representation. Claim 73 contains new additional explanatory text. The applicant respectfully submits that this claim 73 should be allowable.

Comment on claims 30-31 and 41-42:

New versions submitted as claims 73-74 and 84-85.

Claims 73 and 74 are dependent upon claim 71 which contains new additional explanatory text and also on claim 44 which contains new limitations.

The concept of placing explicit dependency information into an instruction set architecture is described on pages 43-44 of the specification. A considerable number of instruction set architectures have been defined in the prior art. Is there a prior-art instruction set architecture that explicitly notes dependencies between instructions using dependency pointers or dependency vectors? If not, then the applicant respectfully submits that the systems described in claims 73, 74, 84, and 85 should not be considered obvious based on the prior art. The relatively large number of prior-art instruction set architectures should constitute evidence of prior-art thinking on the design of instruction sets. The applicant respectfully submits that claims 73, 74, 84, and 85 should be allowable.

Comment on claim 38: New version submitted as claim 81. Claim 81 describes the idea of committing execution results as though the process executed the original instruction set architecture instructions one at a time, allowing interrupts to occur in between instructions. In particular, the results from executing a group of one or more transformed instructions corresponding to a single original instruction set architecture instruction should be committed as a single entity. Ireton presents no mechanism for managing this even though Ireton's system likely will generate multiple atomic operations for a single original instruction in many cases and then feed these atomic operations into a scheduler that may reorder them. It is not clear how committing the results as a single entity will work without a defined mechanism for managing this. The applicant respectfully submits that it is not an obvious extension of Ireton to develop a system as in claim 81. The applicant respectfully submits that claim 81 should be allowable.

Comment on claims 33-37: New versions submitted as claims 76-80. The history tables that are claimed in claims 76-80 are described in pages 45-47 of the specification. These tables can be used effectively with semi-dynamic code re-writing and re-scheduling also presented in this specification on pages 44-47. Claims 77-80 refer to specific types of tables that are not described by Ireton. These history tables record different information and are quite distinct from the idea of using a branch history table cited in the office action. Claims 77-80 have been new explanatory text. The applicant respectfully submits that claims 76-80 should be allowable in their new form because these types of tables are not described in Ireton.

Comment on claim 39 in view of Farber (US Patent 6,105,124): New version submitted as claim 82. Methods of supporting precise interrupts are described on pages 50-52 of the present specification. Because instruction sequences are often reordered by the scheduling process during instruction stream transformation, the assignment of sequence numbers to the transformed instructions is the claimed invention's mechanism of keeping track of the original program order in which to commit instructions. Farber describes that the hyper-block will be scheduled using a software algorithm but provides no information on how to maintain state to support precise interrupts. The examiner points out that it is desirable in conventional instruction set architectures to commit instructions in order.

However, Farber does not present any mechanism for committing instructions in the original program order. As described the Farber system would likely reorder instructions compared to the original program order. Unless a specific mechanism is proposed to commit the instruction results in the original program order, then the Farber system will not be able to maintain precise interrupts.

The applicant respectfully submits that Farber does not describe a mechanism that is close to the limitations of claim 82 and that it is not an obvious extension of Farber. Therefore, the applicant respectfully submits that claim 82 should therefore be allowable.

Comment on claim 43 in view of Farber (US Patent 6,105,124): New version submitted as claim 86. Claim 86 contains additional explanatory text and limitations. Farber did not describe a special instruction stream cache or its functionality. Instead, the Farber system stores its results in main memory. The applicant respectfully submits that claim 86 should be allowable in view of the above explanation and added limitations.

The applicant respectfully submits the new set of claims for examination. The applicant respectfully submits that the new claim set should be allowable based on the reasons above.

The applicant has reviewed the other references cited. These references do not appear to describe the presently claimed inventions and do not appear render the claimed inventions to be obvious in any way.

The applicant is available at telephone number 408-927-7940 to discuss this application.

Thank you for your review of this application and thank you for your help.

Very respectfully,

Derek Wong

Derek Wong

Patent Applicant

Express Mail Label # EU183174322US Date of Deposit March 20, 2002

I hereby certify that this paper or fee is being deposited with the United States Postal Service using "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to "Assistant Commissioner for Patents, Washington, DC 20231."

Signed Derek Wong Inventor